SIGIR18 Papers about Recommendation

Wu, 2019/5/14

Recall

- Online recommendation(1)
- Recommendation with Social Networks(2+1)
  - Group representation, community detection, sequence-aware Rec
- Recommendation with Knowledge Base(1)
- Improve traditional methods(3)
  - APR, CMN, Bandit problem
- Some specific tasks(5)
  - Recommend email, mention, citation, Wikipedia article section
  - Conversational recommender system
- User modeling: Geo-social based(1)

Outline

- Memory Networks
- Recommendation with Knowledge Base
- CMN: Collaborative Memory Network for Recommendation
- APR: Adversarial Personalized Ranking for Recommendation

A Question Answer(QA) Scenario

- Memory:

Joe went to the kitchen
Fred went to the kitchen
Joe picked up the milk
Joe travelled to the office
Joe left the milk
Joe went to the bathroom

- QA:
  - Where is the milk now? A: office
  - Where is Joe? A: bathroom
  - Where was Joe before the office? A: kitchen

Memory Networks

- A memory **m**, and four components I, G, O and R.
  - I: input feature map
  - G(generalization): updates old memories given the new input
  - O: output feature map
  - R(response): converts the output feature into response format desired

Weston J, Chopra S, Bordes A. Memory networks[J]. arXiv preprint arXiv:1410.3916, 2014.

Memory Networks

- A general flow when given an input x:
  - Convert x in to an feature representation $I(x)$.
  - Update memory **m**: $m_i = G(m_i, I(x), m), for\ any\ i$
  - Compute output features o: $\mathrm{o} = \mathrm{O}(\mathrm{I}(\mathrm{x}), \mathrm{m})$
  - Decode o to give final response: $r = R(o)$

A Memory Neural Network for QA

- I & G modules: stores the text in the next available space
  - $m_N = x, N = N + 1$

Joe went to the kitchen
Fred went to the kitchen
Joe picked up the milk
Joe travelled to the office
Joe left the milk
Joe went to the bathroom

A Memory Neural Network for QA

- O module produces output features by finding k supporting memories given input x

$$o_1 = O_1(x, \mathbf{m}) = \underset{i=1,\dots,N}{\arg\max}\ s_O(x, \mathbf{m}_i)$$

$$o_2 = O_2(x, \mathbf{m}) = \underset{i=1,\dots,N}{\arg\max}\ s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_i)$$

- Example: given question ‘Where is the milk now? ’
    - m_{o1}： Joe left the milk
    - $\mathbf{m}_{o2}$: Joe travelled to the office

A Memory Neural Network for QA

- R module returns a textual response r(a word here).

$$r = \operatorname{argmax}_{w \in W} \, s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], w)$$

  - Example: r = office

- $s_o = s_R = s(x, y)$:

$$s(x, y) = \Phi_x(x)^\top U^\top U \Phi_y(y).$$

- Every word in the dictionary has three different representations: one for $\Phi_y(.)$ and two for $\Phi_x(.)$ for query and memory.

A Memory Neural Network for QA

- How to train model: a fully supervised setting

$$\sum_{\bar{f} \neq \mathbf{m}_{o_1}} \max(0, \gamma - s_O(x, \mathbf{m}_{o_1}) + s_O(x, \bar{f})) +$$
$$\sum_{\bar{f}' \neq \mathbf{m}_{o_2}} \max(0, \gamma - s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_{o_2}]) + s_O([x, \mathbf{m}_{o_1}], \bar{f}'])) +$$
$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], r) + s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], \bar{r}]))$$

- Problem: need too many supervising information

End-To-End Memory Networks

- I and G: A memory pool

m
Joe went to the kitchen
Fred went to the kitchen
Joe picked up the milk
Joe travelled to the office
Joe left the milk
Joe went to the bathroom

c
Joe went to the kitchen
Fred went to the kitchen
Joe picked up the milk
Joe travelled to the office
Joe left the milk
Joe went to the bathroom

- **m**: input memory or key memory
- **c**: output memory or value memory or external memory

Sukhbaatar S, Weston J, Fergus R. End-to-end memory networks[C]//Advances in neural information processing systems. 2015: 2440-2448.

End-To-End Memory Networks

- O: given a query **q**, read out representation **u** with attention

$$\text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$$

$$p_i = \text{Softmax}(u^T m_i).$$

$$o = \sum_i p_i c_i.$$

- p_i can be as the similarity between (key) memory m_i and query u

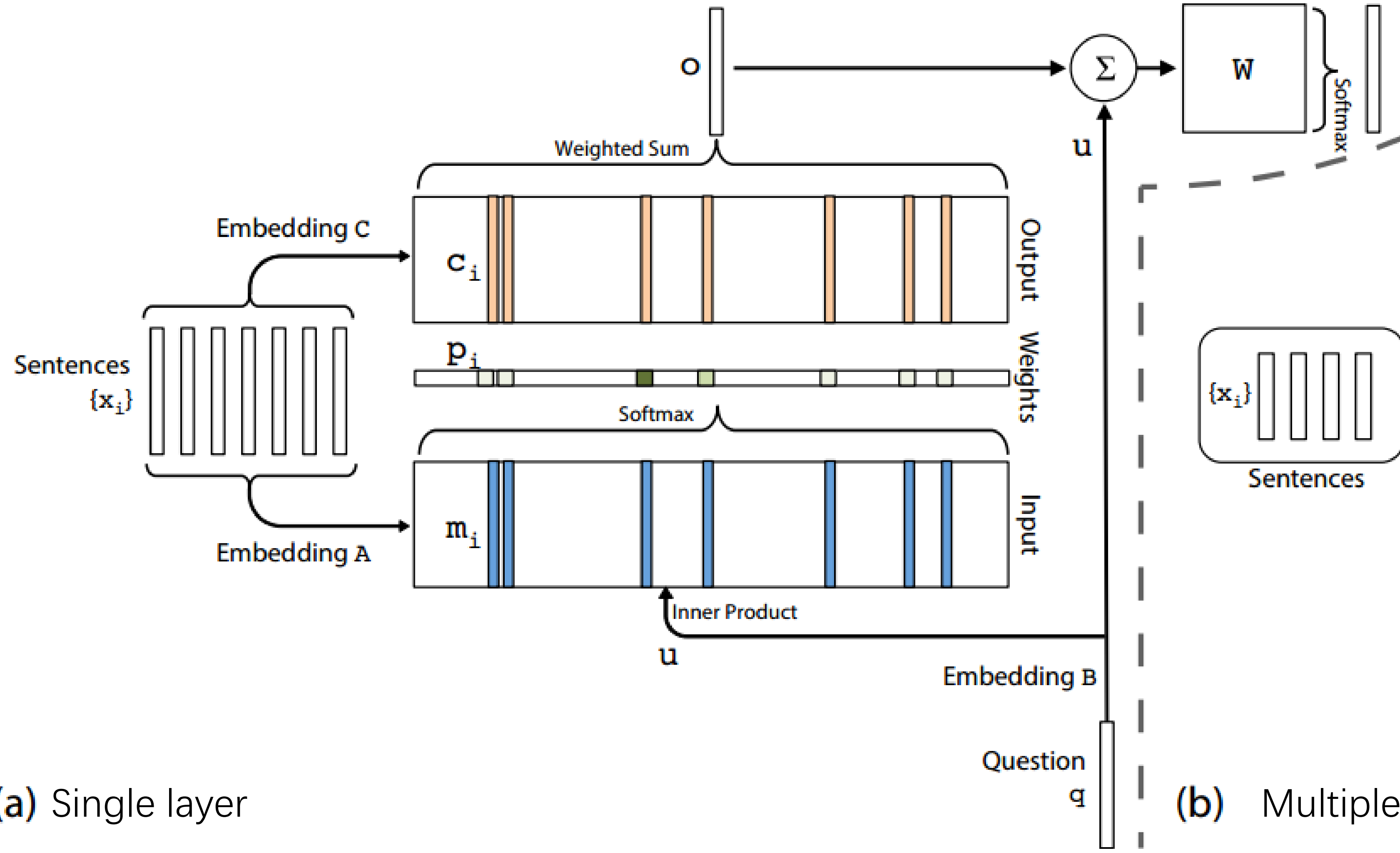
End-To-End Memory Networks

- R: give an answer

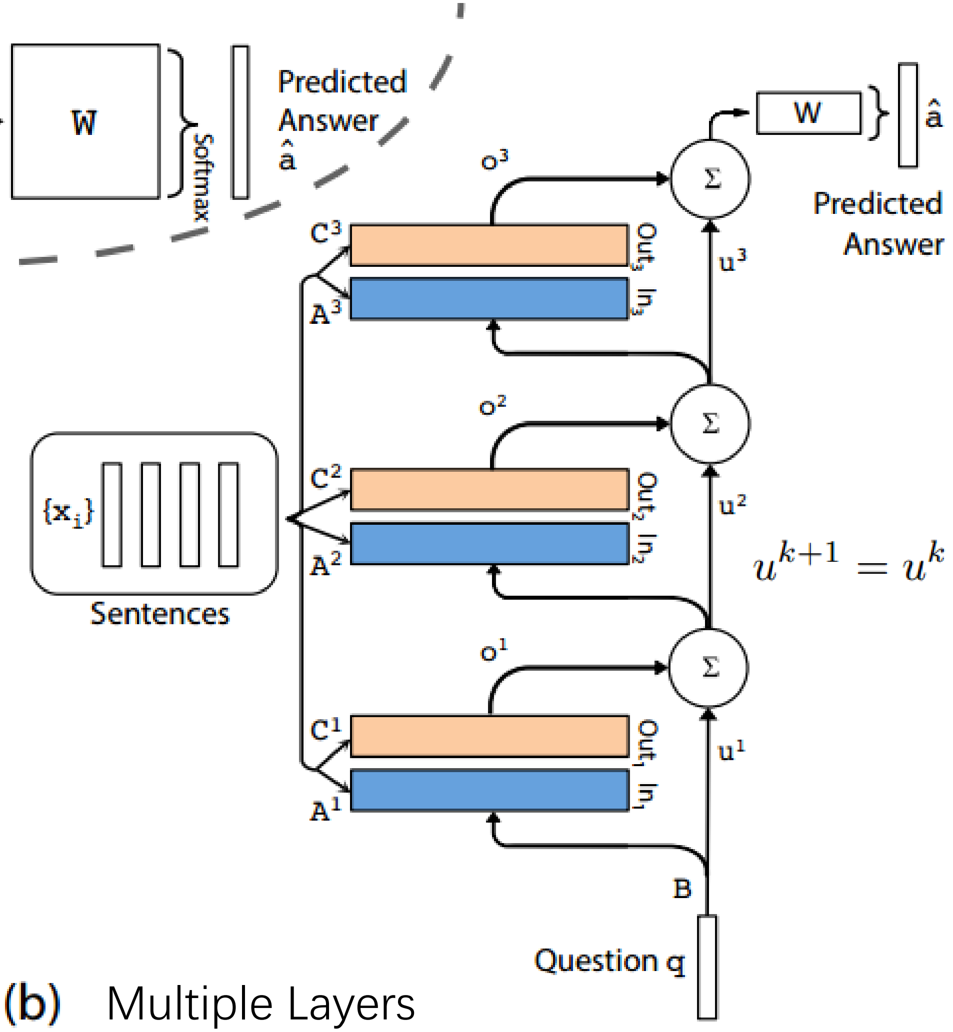
$$\hat{a} = \text{Softmax}(W(o + u))$$

- Loss: L(a, $\hat{a}$)
- Train: end to end, no internal supervising

End-To-End Memory Networks



(a) Single layer



$$u^{k+1} = u^k + o^k.$$

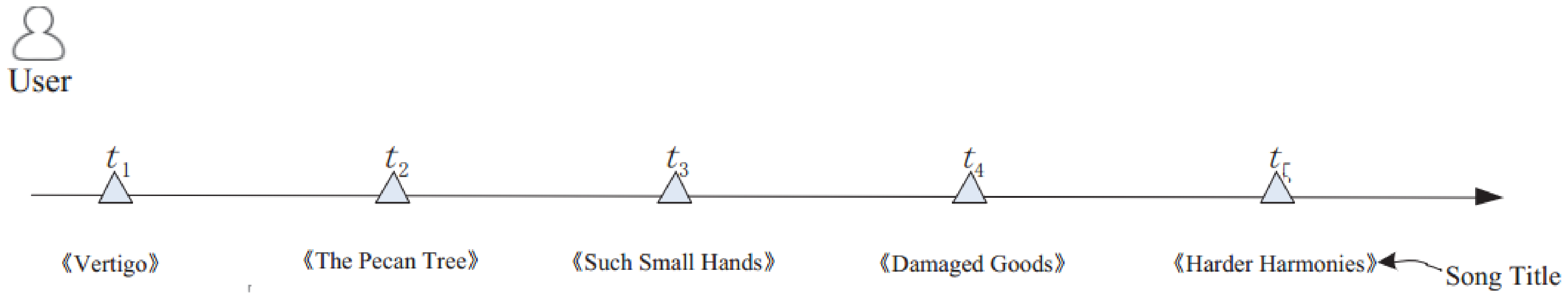
(b) Multiple Layers

Outline

- Memory Networks
  - I, G, O, R
- Recommendation with Knowledge Base
- CMN: Collaborative Memory Network for Recommendation
- APR: Adversarial Personalized Ranking for Recommendation

A Sequential Recommendation Scenario

- Given the interaction sequence of user u, we would like to infer the item that user u will interact with at next time
- An example



Next song?

A GRU-based Sequential Recommender

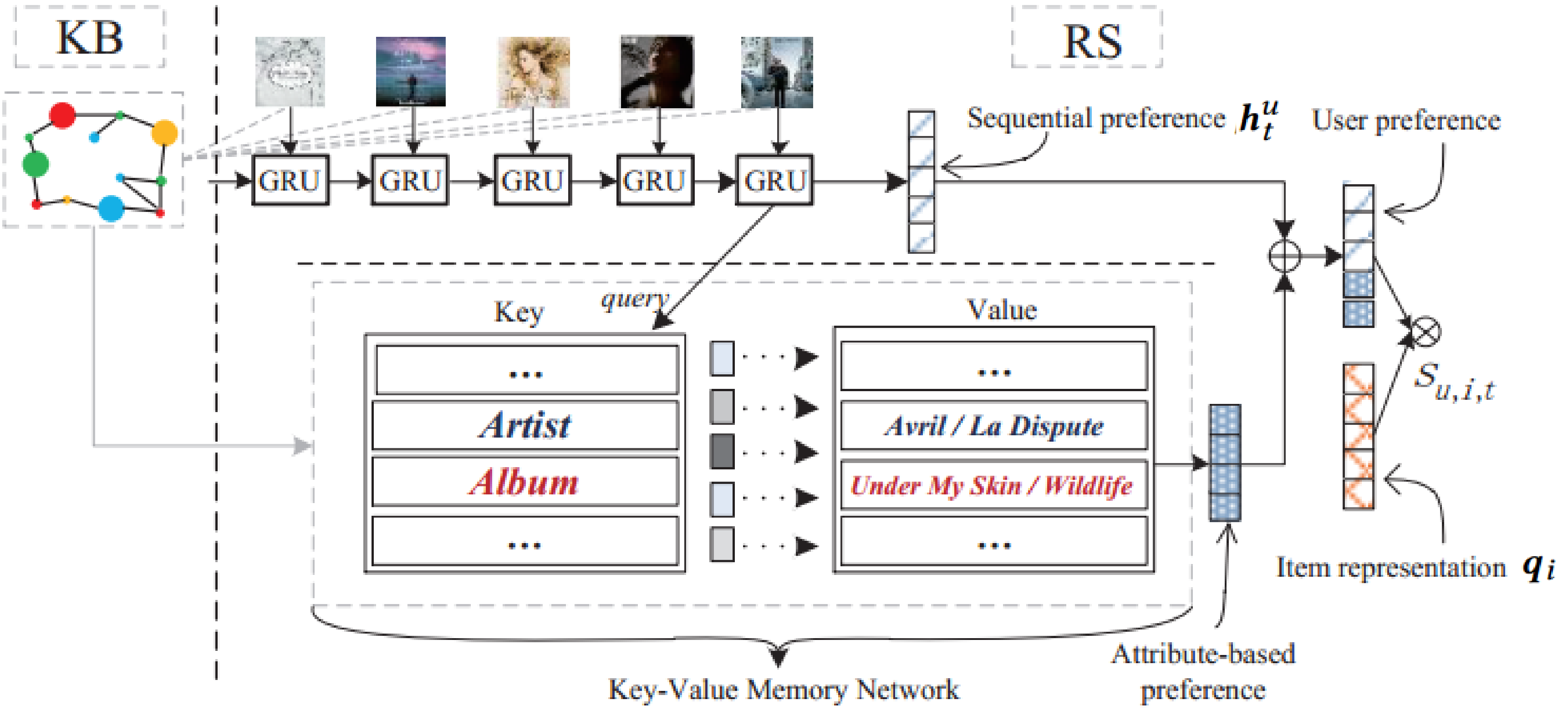
- Sequential preference representation of user u at time t:

$$\boldsymbol{h}_t^u = \mathrm{GRU}(\boldsymbol{h}_{t-1}^u, \boldsymbol{q}_{i_t}; \Theta)$$

- Predict the score of next item i:

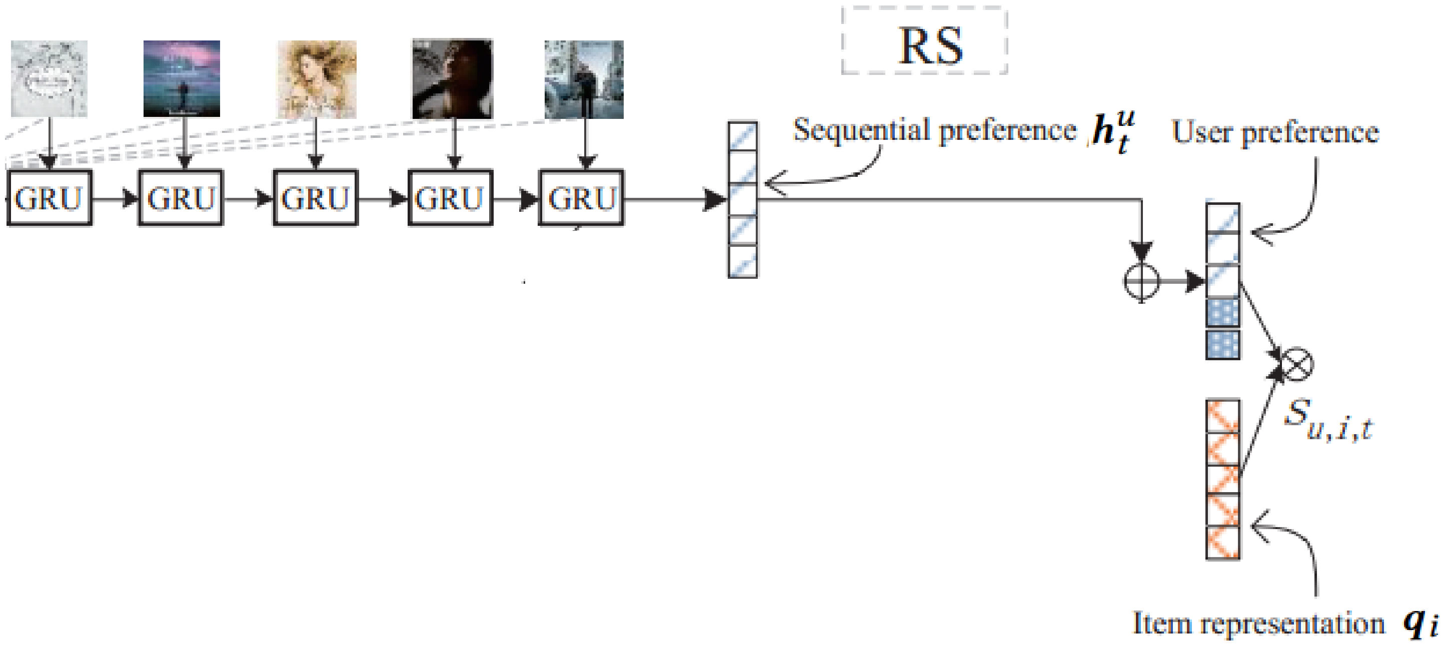
$$s_{u,i,t} = g(u, i, t) = {\boldsymbol{h}_t^u}^\top \cdot \boldsymbol{q}_i,$$

Add Knowledge Base Information

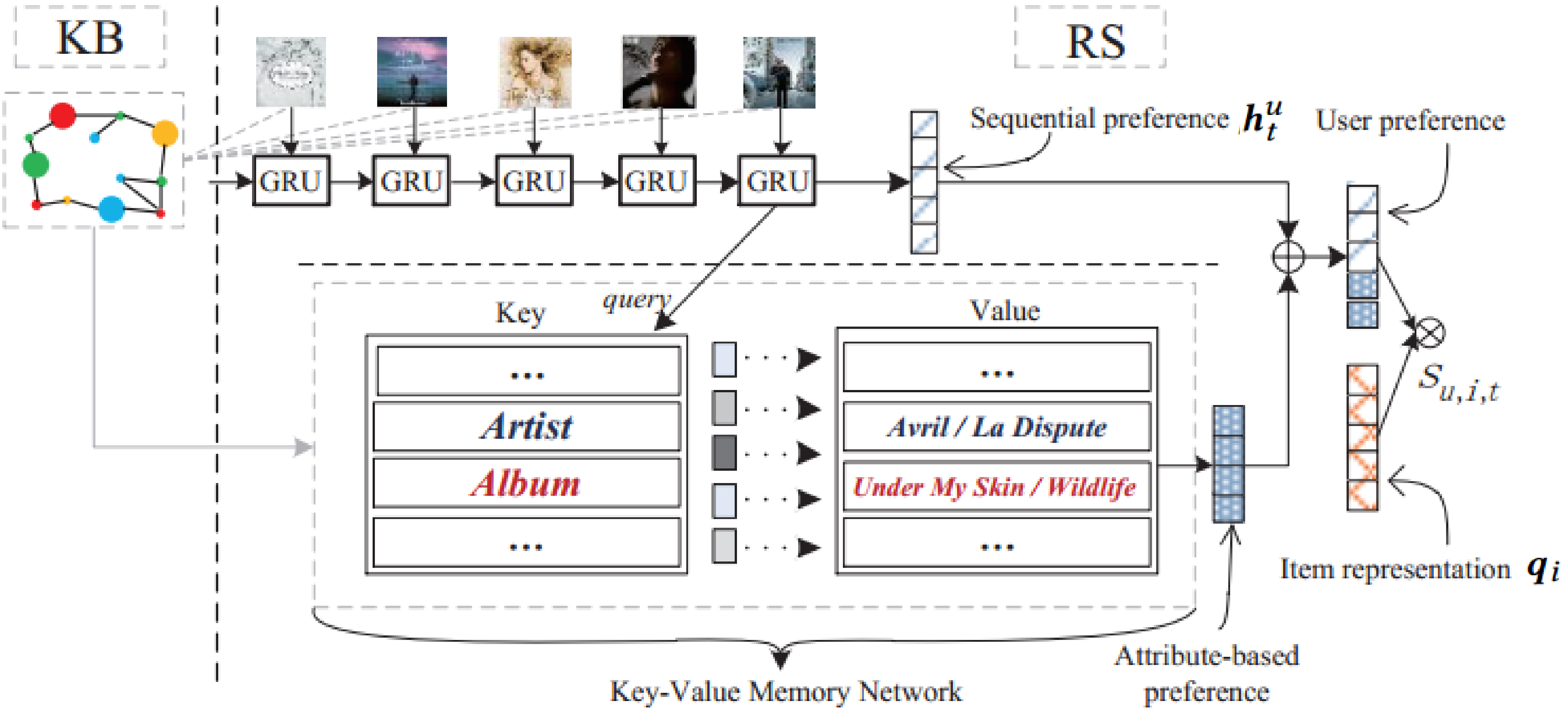


Huang J, Zhao W X, Dou H, et al. Improving sequential recommendation with knowledge-enhanced memory networks[C]//The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM, 2018: 505-514.

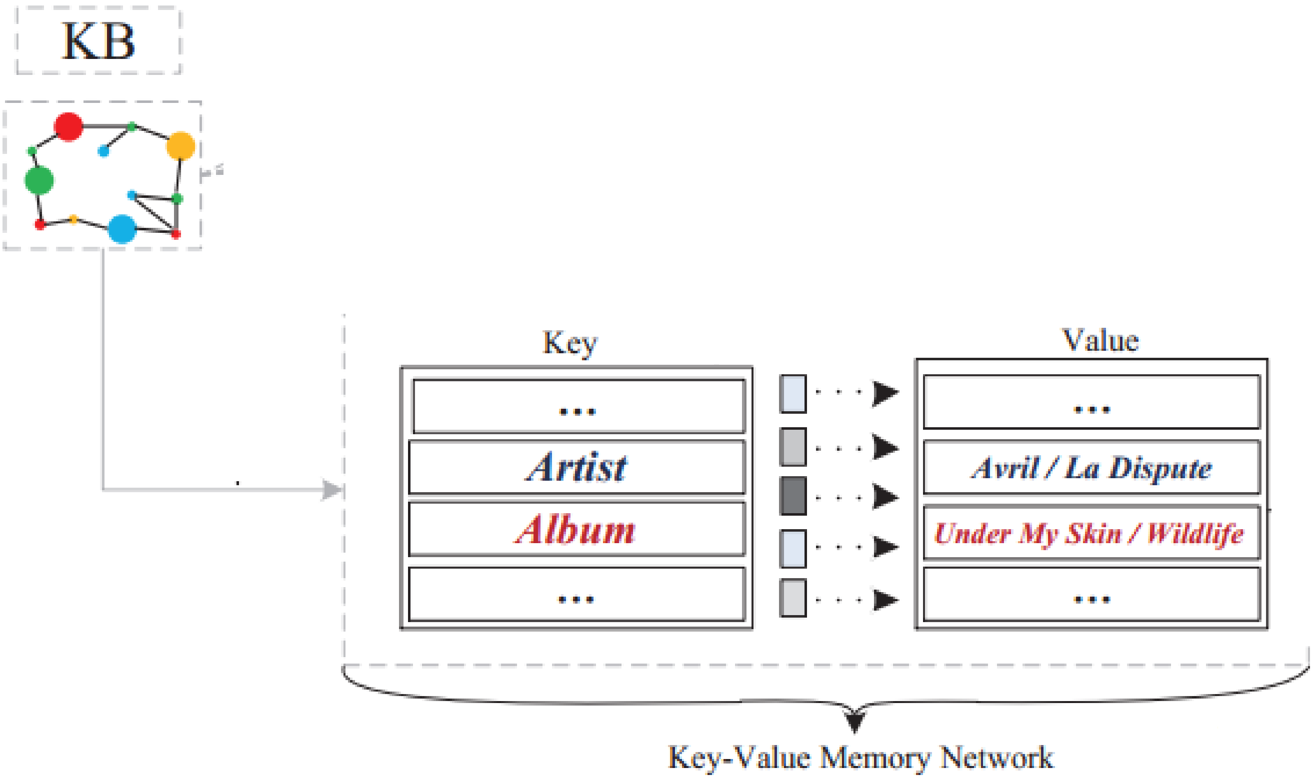
Add Knowledge Base Information



Add Knowledge Base Information



Build Memory Network using KB



Build Memory Network using KB

- In KB, an item has A kinds of attribute information

$$\{(\boldsymbol{k}_1, \boldsymbol{v}_1^u), \cdots, (\boldsymbol{k}_A, \boldsymbol{v}_A^u)\}$$

- I: How to represent the key and value: TransE
  - e_1: item, a song
  - r: attribute, like singer
  - e_2: attribute value, like Adala

$$\sum_{\{\langle e_1, r, e_2\rangle\}} \| \boldsymbol{e}_1 + \boldsymbol{r} - \boldsymbol{e}_2 \|$$

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational

Build Memory Network for a User

- G: use embeddings of attributes as key memory
- G: update value memory. Write operation for a new item i

$$\{\boldsymbol{v}_1^u, \cdots, \boldsymbol{v}_A^u\}^{new} \leftarrow \textsc{Write}(\{(\boldsymbol{k}_1, \boldsymbol{v}_1^u), \cdots, (\boldsymbol{k}_A \boldsymbol{v}_A^u)\}^{old}, \underline{\boldsymbol{e}_i}),$$

- The update value of attribute a is calculated as:

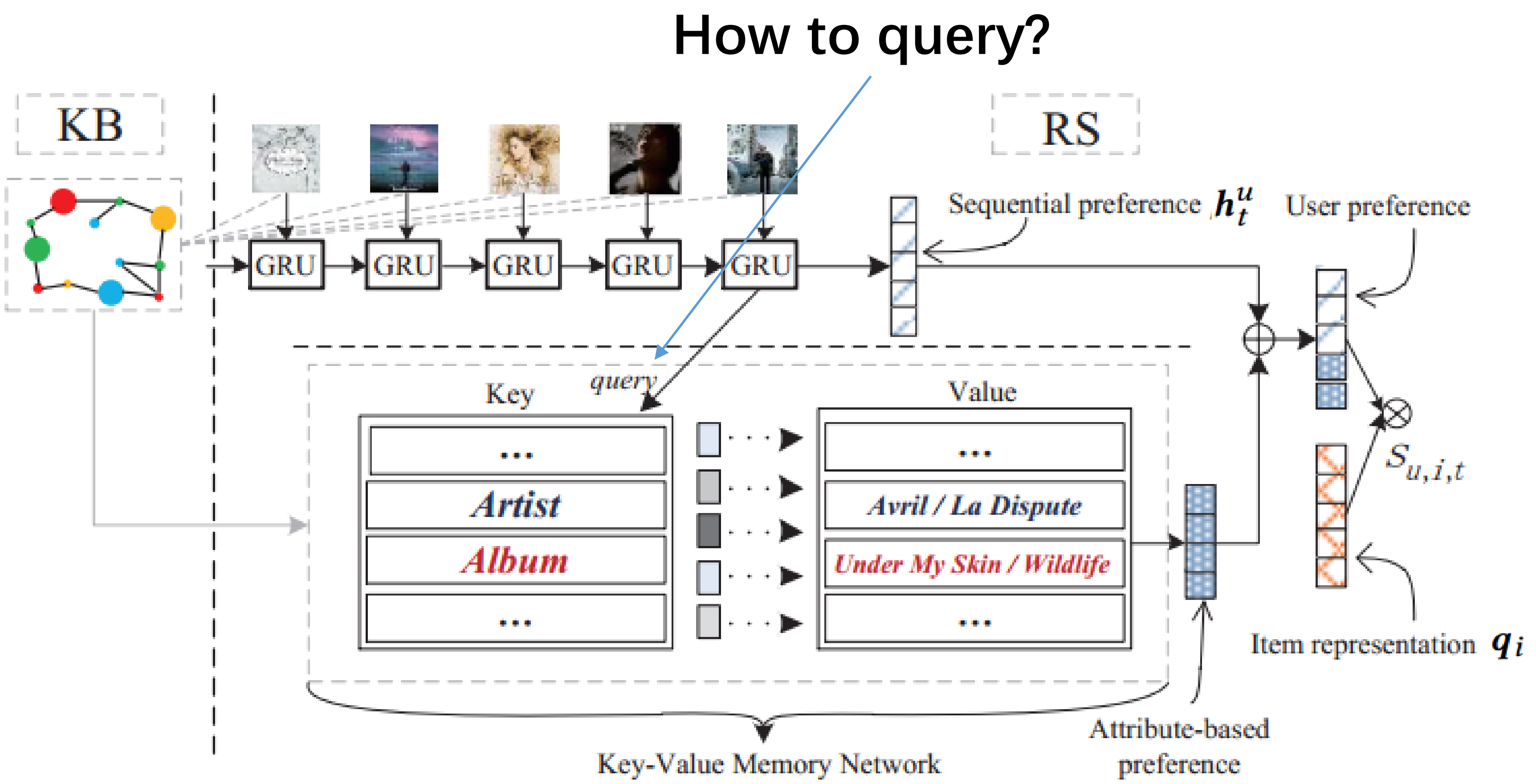
$$\boldsymbol{e}_a^i = \boldsymbol{e}_i + \boldsymbol{r}_a,$$

- Update with gate:

$$z_a = \mathrm{sigmoid}(\boldsymbol{v}_a^{u\top} \cdot \boldsymbol{e}_a^i).$$

$$\boldsymbol{v}_a^u \leftarrow (1 - z_a) \cdot \boldsymbol{v}_a^u + z_a \cdot \boldsymbol{e}_a^i.$$

Using Memory Network Information



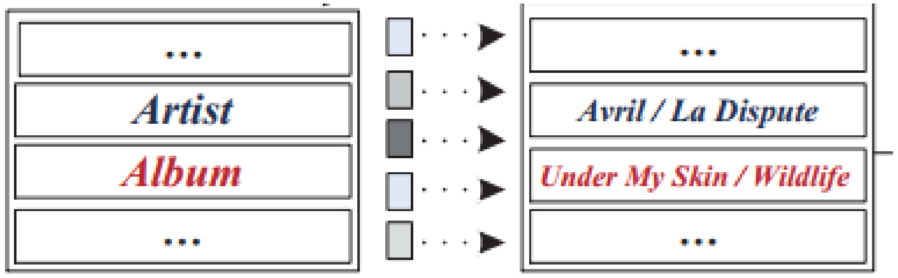
Using memory network info

- O: Read operation

$$\tilde{\boldsymbol{h}}_t^u = \mathrm{MLP}(\boldsymbol{h}_t^u)$$

$$\boldsymbol{m}_t^u \leftarrow \textsc{Read}(\{(\boldsymbol{k}_1, \boldsymbol{v}_1^u), \cdots, (\boldsymbol{k}_A, \boldsymbol{v}_A^u)\}, \underline{\tilde{\boldsymbol{h}}_t^u}),$$

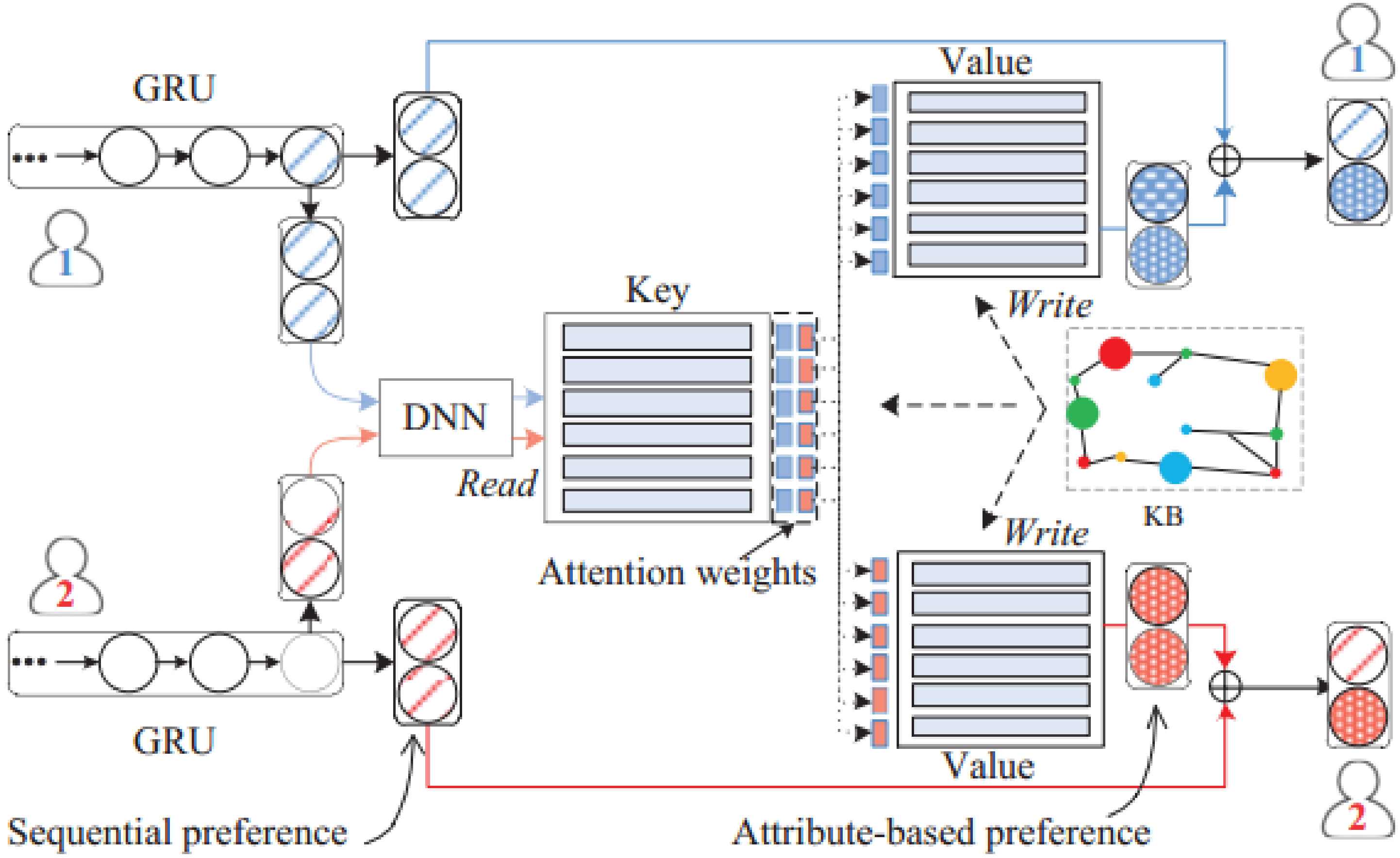
- Read with attention:



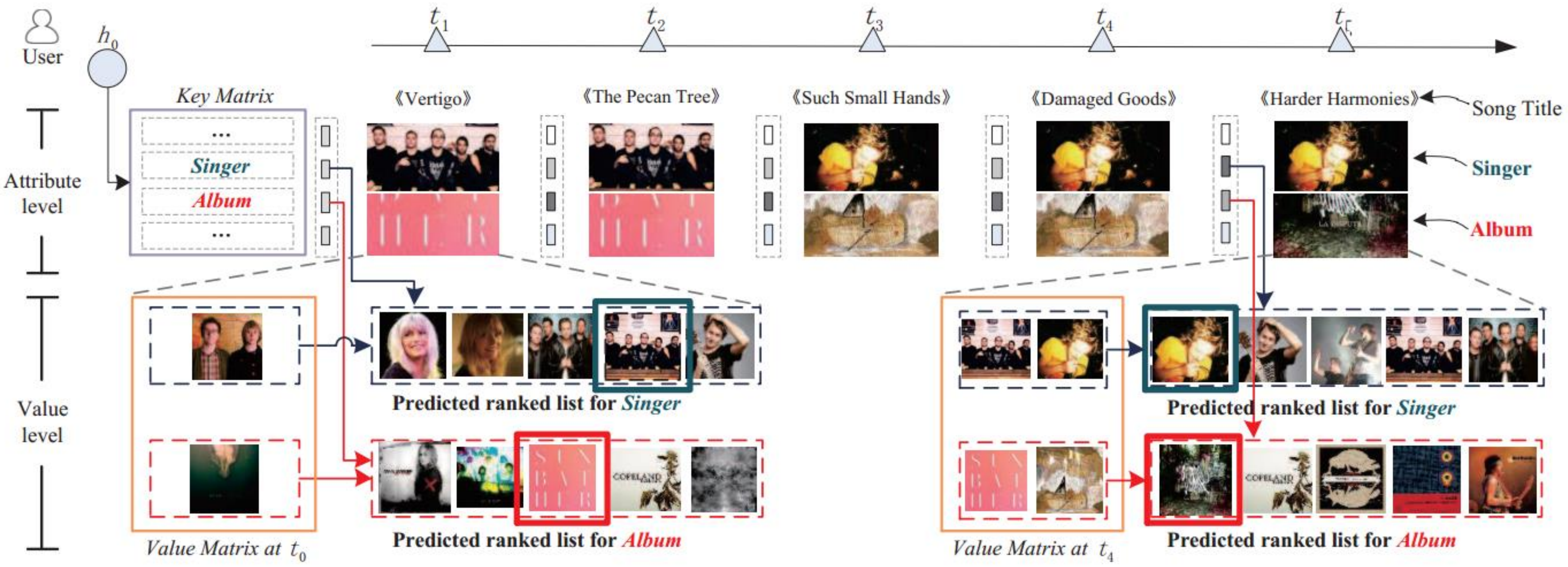
$$\boldsymbol{m}_t^u \leftarrow \sum_{a=1}^{A} w_{t,u,a} \cdot \boldsymbol{v}_a^u,$$

$$w_{t,u,a} = \frac{\exp(\gamma \tilde{\boldsymbol{h}}_t^u \cdot \boldsymbol{k}_a)}{\sum_{a'=1}^{A} \exp(\gamma \tilde{\boldsymbol{h}}_t^u \cdot \boldsymbol{k}_{a'})},$$

Overall architecture



Experiments



Outline

- Memory Networks
  - I, **G**, **O**, R
- Recommendation with Knowledge Base
  - Key-Value Memory Network
- CMN: Collaborative Memory Network for Recommendation
- APR: Adversarial Personalized Ranking for Recommendation

Collaborative Memory Network for Recommendation Systems

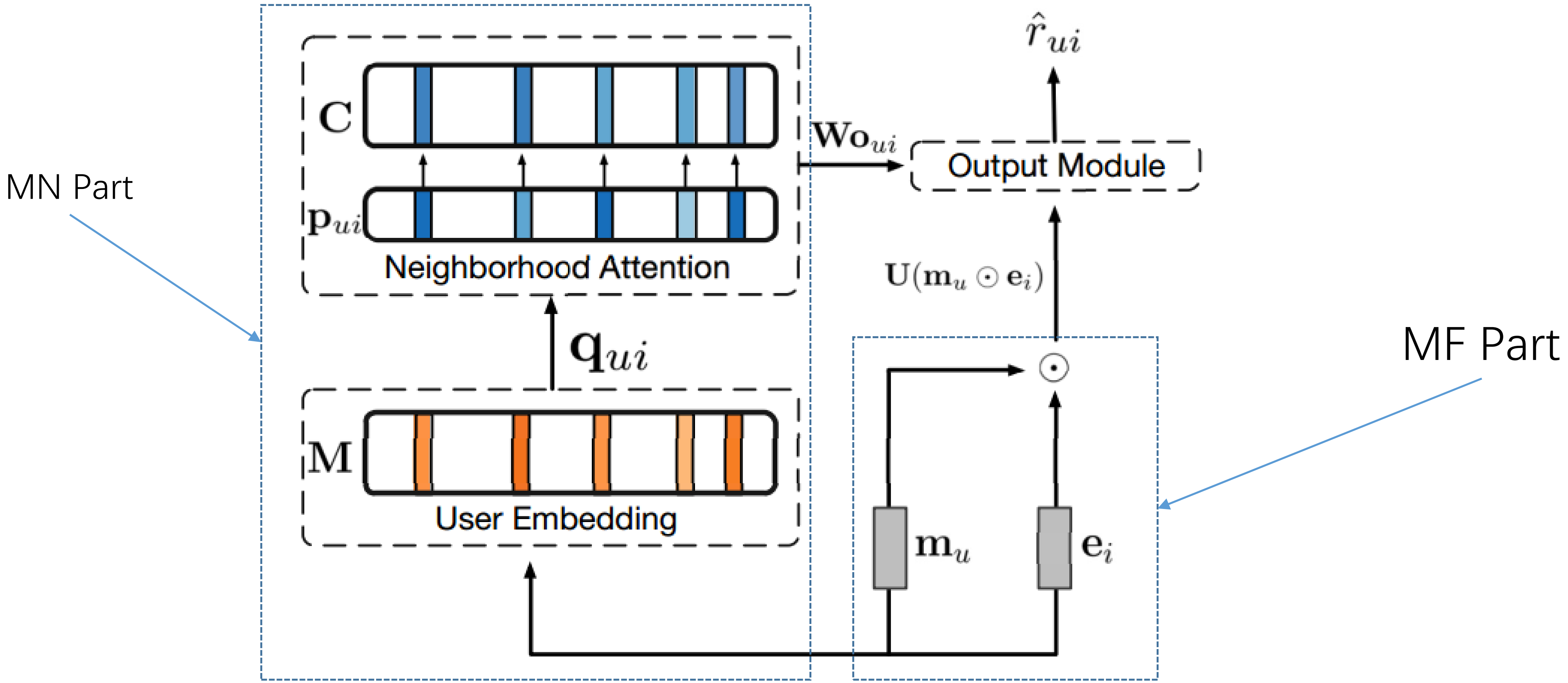
- Motivation:
  - Three categories of collaborative filtering (CF) methods
    - Memory or neighborhood-based methods, like KNN
    - Latent factor models, like Matrix Factorization
    - Hybrid models, like Factorization Machines
  - Neighborhood methods capture local structure.
  - Latent factor models capture the overall global structure of the user and item relationships

- Unify Memory Networks and neural attention mechanisms for CF

Ebesu T, Shen B, Fang Y. Collaborative memory network for recommendation systems[C]//The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM, 2018: 515-524.

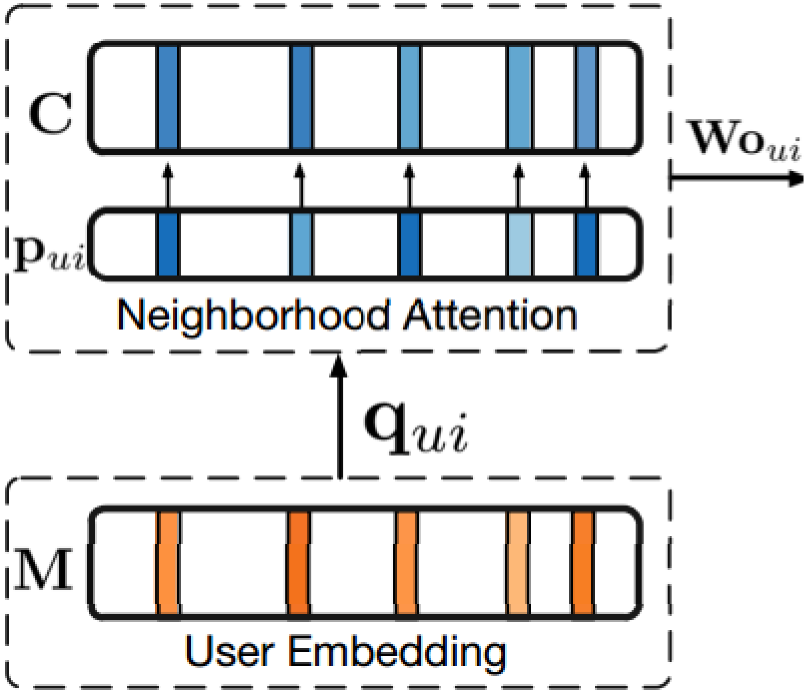
Collaborative Memory Network

- User memory matrix M, Item memory E,



Memory Network Part

- For a user, fetch the preference of his neighbors
- I & G: preference of neighbors is an external embedding matrix **C**
- O: for a query user-item pair (u, i), read with attention



$$\mathbf{o}_{ui} = \sum_{v \in N(i)} p_{uiv} \mathbf{c}_v$$

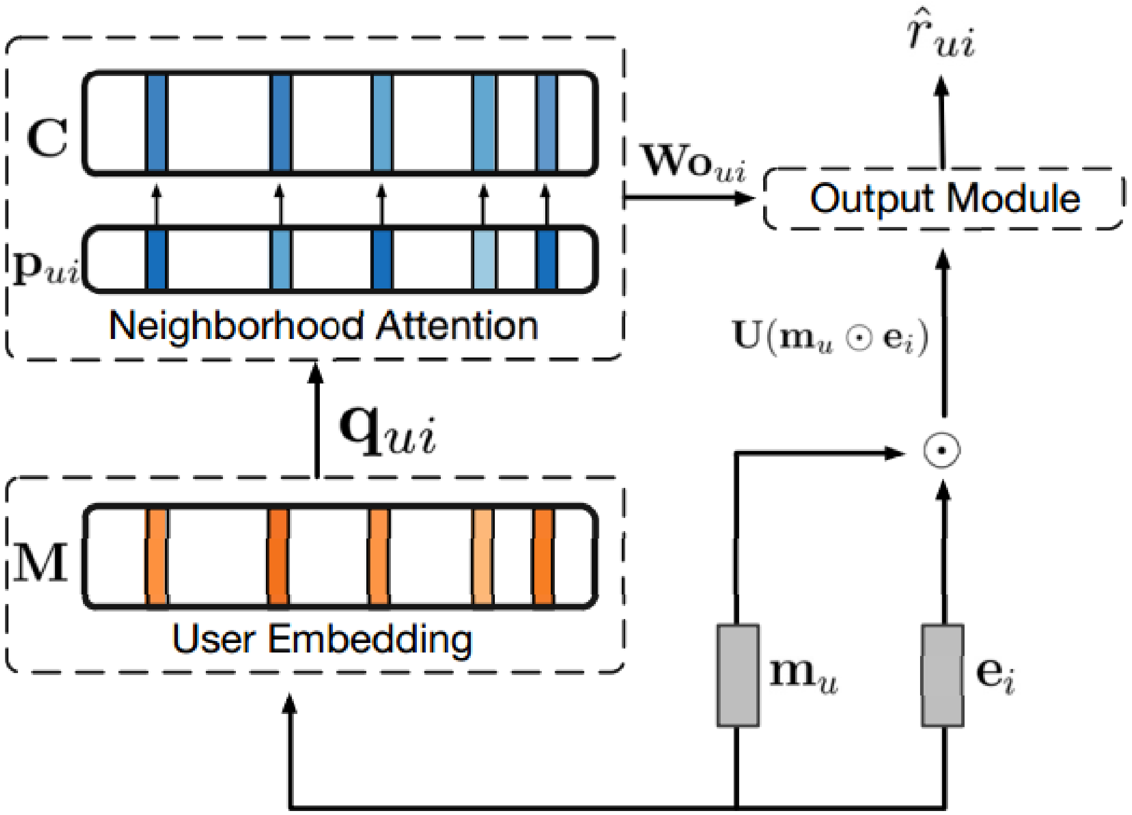
$$q_{uiv} = \mathbf{m}_u^{\mathsf{T}} \mathbf{m}_v + \mathbf{e}_i^{\mathsf{T}} \mathbf{m}_v \quad \forall\, v \in N(i)$$

$$p_{uiv} = \frac{\exp(q_{uiv})}{\sum_{k \in N(i)} \exp(q_{uik})} \quad \forall\, v \in N(i)$$

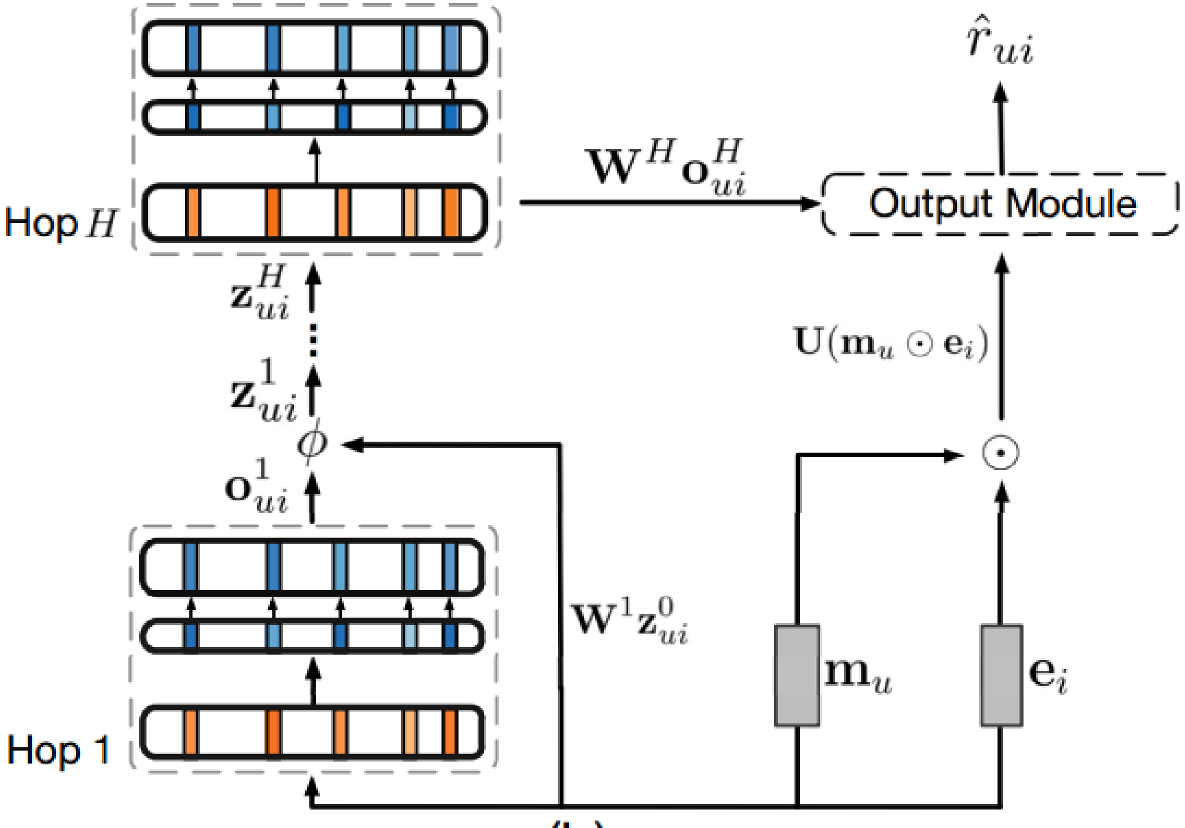
Output Module

- R: output the final predicted score

$$\hat{r}_{ui} = \mathbf{v}^{\mathsf{T}}\phi\big(\mathbf{U}(\mathbf{m}_u \odot \mathbf{e}_i) + \mathbf{W}\mathbf{o}_{ui} + \mathbf{b}\big)$$



Multi Hops



(b)

Query vector z at h layer

$$\mathbf{z}_{ui}^h = \phi(\mathbf{W}^h\mathbf{z}_{ui}^{h-1} + \mathbf{o}_{ui}^h + \mathbf{b}^h)$$

$$\mathbf{z}_{ui}^0 = \mathbf{m}_u + \mathbf{e}_i.$$

New weight

$$q_{uiv}^{h+1} = (\mathbf{z}_{ui}^h)^\top \mathbf{m}_v \quad \forall\, v \in N(i)$$

Outline

- Memory Networks
  - I, **G**, **O**, R
- Recommendation with Knowledge Base
  - Key-Value Memory Network built from KB
- CMN: Collaborative Memory Network for Recommendation
  - External memory matrix for users

- APR: Adversarial Personalized Ranking for Recommendation

Adversarial Personalized Ranking for Recommendation

- Motivation: Optimizing MF with BPR leads to a recommender model that is **not robust**

- The resultant model is highly vulnerable to adversarial perturbations on its model parameters, which **implies the possibly large error in generalization**

- To enhance the **robustness** of a recommender model and thus improve its **generalization performance**

He X, He Z, Du X, et al. Adversarial personalized ranking for recommendation[C]//The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM, 2018: 355-364.

Adversarial Noises

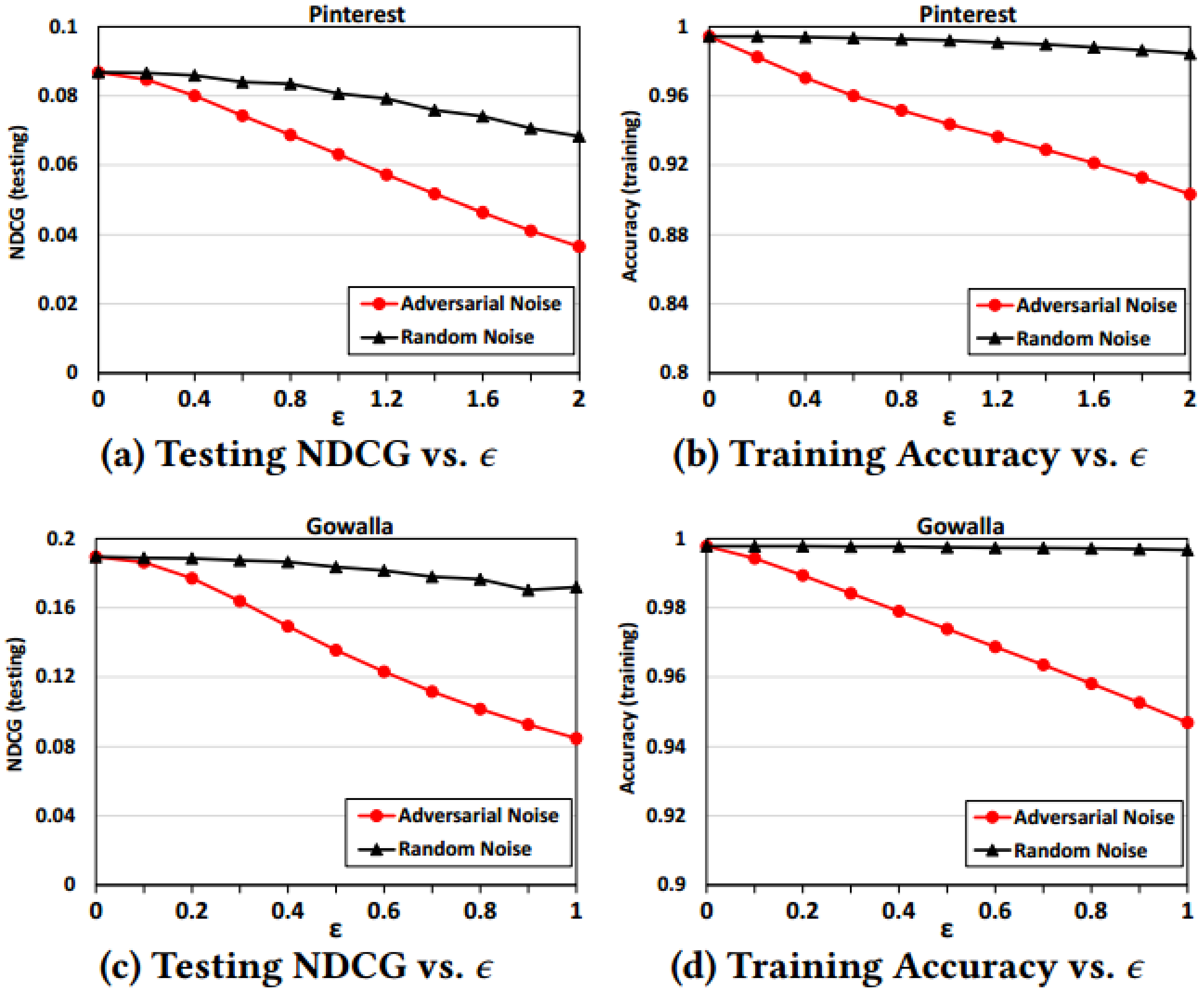
- Defined as the perturbations that aim to maximize the objective function of BPR

$$\Delta_{adv} = \arg \max_{\Delta, ||\Delta|| \leq \epsilon} L_{BPR}(\mathcal{D}|\hat{\Theta} + \Delta),$$

- Approximation

$$\Delta_{adv} = \epsilon \frac{\Gamma}{||\Gamma||} \quad \text{where} \quad \Gamma = \frac{\partial L_{BPR}(\mathcal{D}|\hat{\Theta} + \Delta)}{\partial \Delta}.$$

BPR-MF is Vulnerable



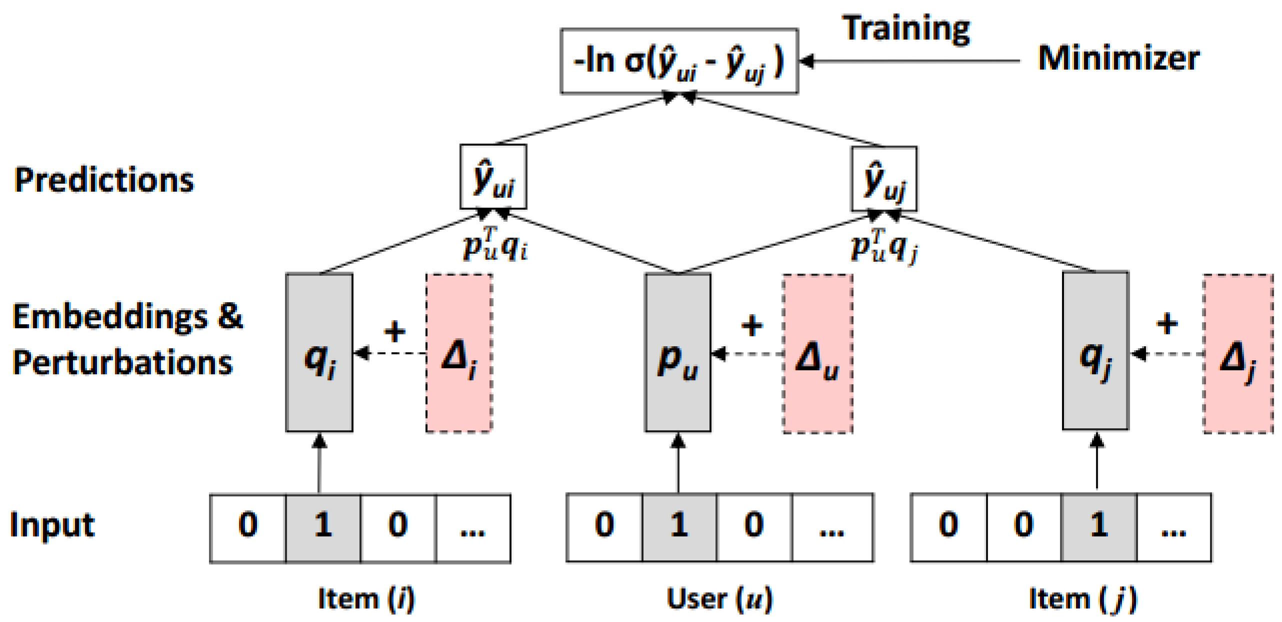
(a) Testing NDCG vs. ϵ

(b) Training Accuracy vs. ϵ

(c) Testing NDCG vs. ϵ

(d) Training Accuracy vs. ϵ

Adversarial Personalized Ranking



$$L_{APR}(\mathcal{D}|\Theta) = L_{BPR}(\mathcal{D}|\Theta) + \lambda L_{BPR}(\mathcal{D}|\Theta + \Delta_{adv}),$$

$$\text{where} \quad \Delta_{adv} = \arg\max_{\Delta, ||\Delta|| \leq \epsilon} L_{BPR}(\mathcal{D}|\hat{\Theta} + \Delta),$$

$$\Theta^*, \Delta^* = \arg\min_{\Theta} \max_{\Delta, ||\Delta|| \leq \epsilon} L_{BPR}(\mathcal{D}|\Theta) + \lambda L_{BPR}(\mathcal{D}|\Theta + \Delta),$$

SGD learning algorithm for APR

Algorithm 1: SGD learning algorithm for APR.

Input: Training data $\mathcal{D}$, adversarial noise level ϵ, adversarial regularizer λ, L_2 regularizer λ_Θ, learning rate η;

Output: Model parameters Θ;

```
Initialize Θ from BPR ;
while Stopping criteria is not met do
    Randomly draw (u, i, j) from D ;
    // Constructing adversarial perturbations
    Δ_adv ← Equation (8) ;
    // Updating model parameters
    Θ ← Equation (11) ;
end
return Θ
```

$$l_{adv}((u,i,j)|\Delta) = -\lambda \ln \sigma(\hat{y}_{ui}(\hat{\Theta}+\Delta) - \hat{y}_{uj}(\hat{\Theta}+\Delta)).$$

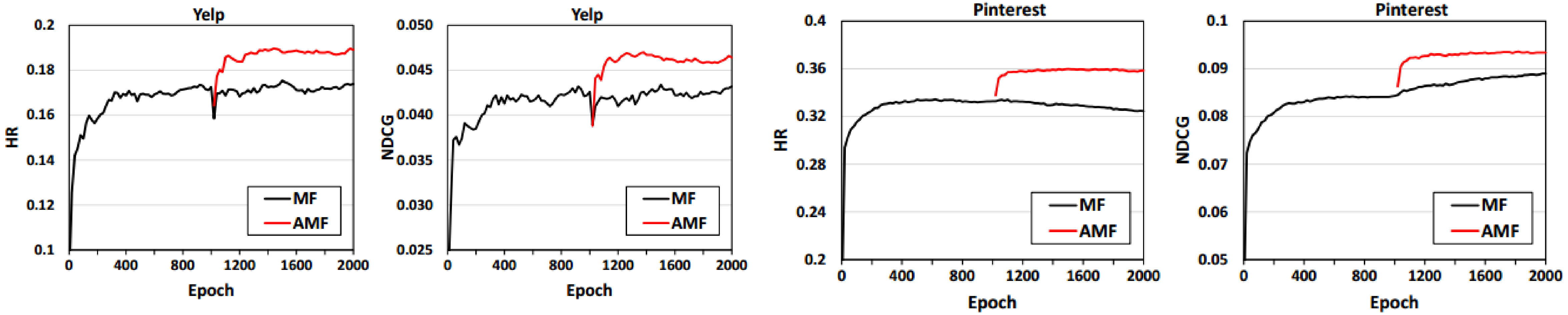
$$\Delta_{adv} = \epsilon \frac{\Gamma}{||\Gamma||} \quad \text{where} \quad \Gamma = \frac{\partial l_{adv}((u,i,j)|\Delta)}{\partial \Delta}. \tag{8}$$

$$\begin{aligned} l_{APR}((u,i,j)|\Theta) = & -\ln \sigma(\hat{y}_{ui}(\Theta) - \hat{y}_{uj}(\Theta)) + \lambda_\Theta ||\Theta||^2 \\ & -\lambda \ln \sigma(\hat{y}_{ui}(\Theta+\Delta_{adv}) - \hat{y}_{uj}(\Theta+\Delta_{adv})). \end{aligned}$$

$$\Theta = \Theta - \eta \frac{\partial l_{APR}((u,i,j)|\Theta)}{\partial \Theta}, \tag{11}$$

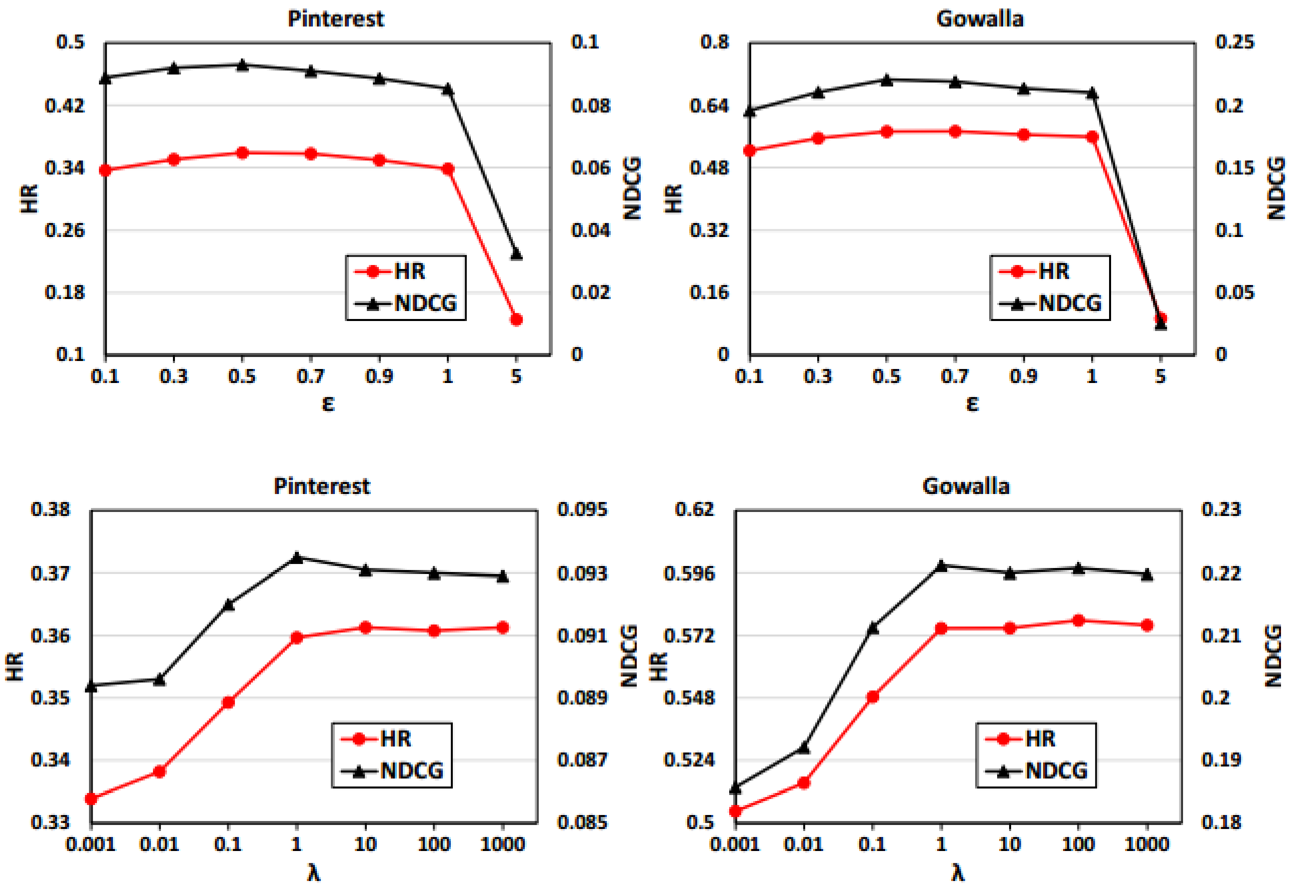
Experiments

RQ1 How is the effect of adversarial learning? Can AMF improve over MF-BPR by performing adversarial learning?



Experiments

RQ3 How do the hyper-parameters ϵ and λ affect the performance and how to choose optimal values?



Thank you